

CLAIMS

WHAT IS CLAIMED:

1. A method, comprising:
5 identifying a loop in a program;
identifying each vector memory reference in the loop
determining dependencies between vector memory references in the loop, including
determining unidirectional and circular dependencies; and
distributing the vector memory references into a plurality of detail loops, wherein the
10 vector memory references that have circular dependencies therebetween are included in a
common detail loop, and wherein the detail loops are ordered according to the unidirectional
dependencies between the memory references.
2. A method, as set forth in claim 1, further comprising allocating a plurality of
15 temporary storage areas within a cache and determining the size of each temporary storage
area based on the size of the cache and the number of temporary storage areas.
3. A method, as set forth in claim 1, further comprising at least one section loop
including the plurality of detail loops.
20
4. A method, as set forth in claim 1, wherein distributing the vector memory
references into a plurality of detail loops further comprises distributing the vector memory
references into a plurality of detail loops that each contain at least one vector memory
reference that could benefit from cache management.

5. A method, as set forth in claim 1, further comprising inserting cache management instructions into at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory.

6. A method, as set forth in claim 1, further comprising inserting prefetch instructions into at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory.

7. A method, as set forth in claim 1, further comprising performing loop unrolling on at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory.

8. A method, as set forth in claim 1, further comprising inserting at least one of a prefetch instruction and a cache management instruction into at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory, and performing loop unrolling on at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory.

9. A method, comprising:
identifying a loop in a program;
identifying each vector memory reference in the loop
determining dependencies between vector memory references in the loop; and

distributing the vector memory references into a plurality of detail loops, wherein the vector memory references that have dependencies therebetween are included in a common detail loop.

5 10. A method, as set forth in claim 9, further comprising allocating a plurality of temporary storage areas within a cache and determining the size of each temporary storage area based on the size of the cache and the number of temporary storage areas.

10 11. A method, as set forth in claim 9, further comprising at least one section loop including the plurality of detail loops.

15 12. A method, as set forth in claim 9, wherein distributing the vector memory references into a plurality of detail loops further comprises distributing the vector memory references into a plurality of detail loops that each contain at least one vector memory reference that could benefit from cache management.

20 13. A method, as set forth in claim 9, further comprising inserting cache management instructions into at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory.

25 14. A method, as set forth in claim 9, further comprising inserting prefetch instructions into at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory.

15. A method, as set forth in claim 9, further comprising performing loop unrolling on at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory.

5 16. A method, as set forth in claim 9, further comprising inserting at least one of a prefetch instruction and a cache management instruction into at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory, and performing loop unrolling on at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and
10 main memory.

17. A method, comprising:
identifying a loop in a program;
identifying each vector memory reference in the loop
15 determining dependencies between vector memory references in the loop; and
distributing the vector memory references into a plurality of detail loops in response to cache behavior and the dependencies between the vector memory references in the loop.

18. A method, as set forth in claim 17, wherein distributing the vector memory
20 references further comprises distributing the vector memory references into the plurality of detail loops with each loop having at least one of the identified vector memory references.

19. A method, as set forth in claim 17, further comprising determining
dependencies between vector memory references in the loop, and wherein distributing the
25 loop includes distributing the vector memory references into the plurality of detail loops,

wherein the vector memory references that have circular dependencies therebetween are included in a common detail loop.

20. A method, as set forth in claim 17, further comprising inserting cache
5 management instructions into at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory.

21. A method, as set forth in claim 17, further comprising inserting prefetch
instructions into at least one of said detail loops to control movement of data associated with
10 the vector memory reference between a cache and main memory.

22. A method, as set forth in claim 17, further comprising performing loop
unrolling on at least one of said detail loops to control movement of data associated with the
vector memory reference between a cache and main memory.

23. A method, as set forth in claim 17, further comprising inserting at least one of
a prefetch instruction and a cache management instruction into at least one of said detail
loops to control movement of data associated with the vector memory reference between a
cache and main memory, and performing loop unrolling on at least one of said detail loops to
15 control movement of data associated with the vector memory reference between a cache and
main memory.

24. A computer programmed to perform a method, comprising:
identifying a loop in a program;

identifying each vector memory reference in the loop

determining dependencies between vector memory references in the loop; and

distributing the vector memory references into a plurality of detail loops, wherein the vector memory references that have circular dependencies therebetween are included in a
5 common detail loop.

25. A program storage medium encoded with instructions that, when executed by a computer, perform a method, comprising:

identifying a loop in a program;

identifying each vector memory reference in the loop

10 determining dependencies between vector memory references in the loop; and

distributing the vector memory references into a plurality of detail loops, wherein the vector memory references that have circular dependencies therebetween are included in a
common detail loop.

15 26. A compiler, comprising:

means for identifying a loop in a program;

means for identifying each vector memory reference in the loop

means for determining dependencies between vector memory references in the loop,
including determining unidirectional and circular dependencies; and

20 means for distributing the vector memory references into a plurality of detail loops,
wherein the vector memory references that have circular dependencies therebetween are included in a common detail loop, and wherein the detail loops are ordered according to the unidirectional dependencies between the memory references .

27. A compiler, as set forth in claim 26, further comprising means for allocating a plurality of temporary storage areas within a cache and determining the size of each temporary storage area based on the size of the cache and the number of temporary storage areas.

28. A compiler, as set forth in claim 26, wherein the means for distributing the vector memory references into a plurality of detail loops further comprises distributing the vector memory references into a plurality of detail loops that each contain at least one vector memory reference that could benefit from cache management.

29. A compiler, as set forth in claim 26, further comprising inserting cache management instructions into at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory.

30. A compiler, as set forth in claim 26, further comprising means for inserting prefetch instructions into at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory.

31. A compiler, as set forth in claim 26, further comprising means for performing loop unrolling on at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory.

32. A compiler, as set forth in claim 26, further comprising means for inserting at least one of a prefetch instruction and a cache management instruction into at least one of

said detail loops to control movement of data associated with the vector memory reference between a cache and main memory, and performing loop unrolling on at least one of said detail loops to control movement of data associated with the vector memory reference between a cache and main memory.

5

33. A method for reducing the likelihood of cache thrashing by software to be executed on a computer system having a cache, comprising:
executing the software on the computer system;
generating a profile indicating the manner in which the software uses the cache;
identifying a portion of the software using the profile data that may experience cache thrashing; and
modifying the identified portion of the software to reduce the likelihood of cache thrashing.

10
15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95
100
105
110
115
120
125
130
135
140
145
150
155
160
165
170
175
180
185
190
195
200
205
210
215
220
225
230
235
240
245
250
255
260
265
270
275
280
285
290
295
300
305
310
315
320
325
330
335
340
345
350
355
360
365
370
375
380
385
390
395
400
405
410
415
420
425
430
435
440
445
450
455
460
465
470
475
480
485
490
495
500
505
510
515
520
525
530
535
540
545
550
555
560
565
570
575
580
585
590
595
600
605
610
615
620
625
630
635
640
645
650
655
660
665
670
675
680
685
690
695
700
705
710
715
720
725
730
735
740
745
750
755
760
765
770
775
780
785
790
795
800
805
810
815
820
825
830
835
840
845
850
855
860
865
870
875
880
885
890
895
900
905
910
915
920
925
930
935
940
945
950
955
960
965
970
975
980
985
990
995

20

34. A method, as set forth in claim 33, wherein modifying the identified portion of the software to reduce the likelihood of cache thrashing further comprises:
identifying a loop in the identified portion of the software;
identifying each vector memory reference in the identified loop;
determining dependencies between the vector memory references in the identified loop of the software, including determining unidirectional and circular dependencies; and
distributing the vector memory references into a plurality of detail loops, wherein the vector memory references that have circular dependencies therebetween are included in a common detail loop, and wherein the detail loops are ordered according to the unidirectional dependencies between the memory references.

35. A method for reducing the likelihood of cache thrashing by software to be executed on a computer system having a cache, comprising:

executing the software on the computer system;

generating a profile indicating the manner in which the memory references of the

5 software use the cache;

identifying a first and second portion of the memory references based on the profile,

wherein the first portion of the memory references may be experiencing cache thrashing; and

distributing at least a portion of the first portion of the memory references into distinct loops, and placing at least the second portion of the memory references into the distinct
10 loops.